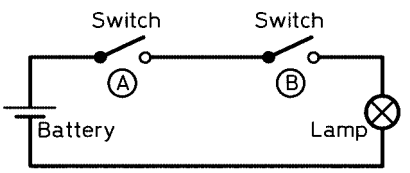
59 Digital logic circuits

Introduction

Logic circuits form the backbone of even the most advanced computer, yet their basic operation can be demonstrated by a couple of switches, a battery and a bulb.

Logic using switches

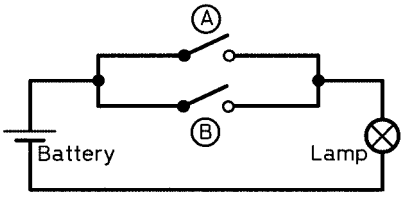
Everyone reading this article will look at **Figure 1** and know immediately how it works and be able to write down something like 'When switch A and switch B are closed, the light will come on'. Without knowing it, you have written down a logic statement involving the so-called **AND** operation; the light comes on only when switches A **AND** B are ON. Below the circuit in Figure 1, is a table showing the only possible positions of the two switches and the state of the bulb for each position. This is called a *truth table*, and is frequently used in logic analysis.



A	B	Lamp
Off	Off	Off
On	Off	Off
Off	On	Off
On	On	On

Figure 1 Switches and lamp AND gate

Figure 2 shows a different circuit. Here, the two switches are in parallel rather than in series, as was the case in Figure 1. Again, if you analyse the circuit in words, you would say that the light will be on when switch A **OR** switch B is ON. This is an example of the **OR** operation, and its truth table is shown in Figure 2. The statement above is not complete, however; can you see why? The truth table will show you. The light comes on if A is ON, **OR** if B is ON, ***OR*** if A **AND** B are both ON. That third condition is easy to miss, but don't worry about it!



A	B	Lamp
Off	Off	Off
Off	On	On
On	Off	On
On	On	On

©RSGB DY184

Figure 2 Switches and lamp OR gate

Believe it or not, some very complicated logic is possible (in theory) using switches and lights, but it is highly impractical and would be very slow. This is where electronic logic circuits come in.

Switches with no moving parts

You may have come across projects in this book where a statement is made such as '. . . the transistor is being used as a switch . . .'. Transistors *can* be used as switches, as were thermionic valves in the world's first programmable computer *Colossus*, at Bletchley Park. However, technology has moved on from valves, through transistors to *logic gates*, combinations of electronic switches designed specifically to perform logic functions.

These act on voltage levels as their inputs and produce changes in voltage levels as their outputs. A positive voltage is called *logic 1*, and corresponds to a switch being ON in our previous descriptions; a zero voltage is called *logic 0*, and corresponds to a switch being OFF. The output from a logic gate (normally labelled Q) is also logic 1 or logic 0, corresponding to our light being ON or OFF, respectively, in our switch analogy.

Many logic devices operate from a stabilised 5 V supply, and this determines the *ideal* voltages corresponding to the two logic states:

logic 0 ≡ 0 V,

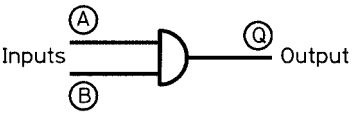
logic 1 ≡ 5 V.

The world isn't an ideal place, so the *real* voltage ranges used by the logic gates are:

logic 0 ≡ 0.0 to 0.4 V,

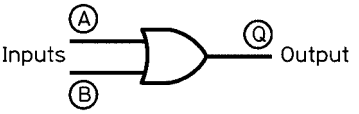
logic 1 ≡ 3.0 to 5.0 V.

The **AND** circuit of Figure 1 is now called an **AND gate**, and requires logic 1 inputs on A **AND** B to produce a logic 1 at the output. **Figure 3** shows this. The truth table is identical with that of Figure 1 – logic 1 replaces ON and logic 0 replaces OFF. Now compare Figure 2 with **Figure 4** – circuits and truth tables for the **OR** function. Again, we have exact similarity.



Inputs		Output
A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

Figure 3 Electronic AND gate

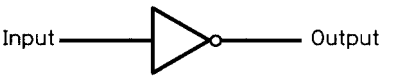


Inputs		Output
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

Figure 4 Electronic OR gate

There is another very common logic gate, which performs the **NOT** function. It is easy to understand. Just ask yourself the question 'What is **NOT** logic 0?', and the answer is obviously 'logic 1'. Similarly, logic 0 is **NOT** logic 1. A **NOT** gate simply changes the logic state of the input; it is also known (because of this behaviour) as an *inverter*. Its symbol and truth table can be found in **Fig 5**. Note the little circle on the output of the gate in Figure 5. In logic circuits, this symbol always implies inversion, or the presence of a **NOT** gate. Keep an eye open for it!

So far, the logic functions we have discussed have all been words which we use in everyday language, which has made the electronic interpretation of them relatively easy. Now we must introduce a function for which there is *no* analogy in normal speech – the **NAND** function. This means a



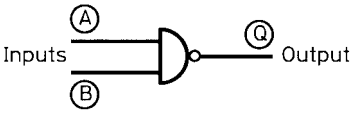
Input	Output
1	0
0	1

©RSGB DY185

Figure 5 Electronic NOT gate

combination of an **AND** gate and a **NOT** gate, and the sharp-eyed reader will have spotted the little circle added on to the normal **AND** gate symbol in **Figure 6**!

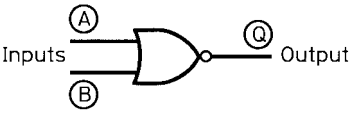
To make things easier to understand, the truth table in Figure 6 has four columns, not three as in previous tables. The third column is the standard **AND** output – compare it with the third column in Figure 3. That is the output from the **AND** gate *before* it encounters the little circle that inverts it, so the final output from the **NAND** gate is the output of the **AND** gate, inverted! The third and fourth columns are the inverse of each other.



Inputs		and	Output
A	B		Q
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

©RSGB DY186

Figure 6 Electronic NAND gate



Inputs		or	Output
A	B		Q
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

©RSGB DY187

Figure 7 Electronic NOR gate

We can add the inverting operation to the output of an **OR** gate also, producing a **NOR** gate! The symbol and its truth table are shown in **Figure 7**. Again, notice that the final output is that of the ordinary **OR** gate, inverted!

These new functions of **NAND** and **NOR** are used more than the **AND** and **OR** functions because it makes other circuits easier to design using combinations of these gates.

A taste of Boolean algebra

The design of circuits using combinations of logic gates usually begins with a little mathematics, where the functions to be implemented are analysed. The mathematics used is surprisingly simple, and is a slightly changed version of ordinary algebra called *Boolean algebra*, which allows manipulation of logic functions to be made. Normal algebra has operations in it such as addition, subtraction and multiplication and division. The mathematician Boole found that the logical **AND** operation could be handled by the algebraic operation of *multiplication* (symbols $\times$ or $\bullet$), and the **OR** operation by the algebraic operation of *addition* (symbol +). The **NOT** operation involved a new symbol, that of a bar over the input being inverted, such as $\overline{A}$.

So, our five basic logic operators can now be written in a mathematical form:

AND	$Q = A \times B$
OR	$Q = A + B$
NOT	$Q = \overline{A}$
NAND	$Q = \overline{A \times B}$
NOR	$Q = \overline{A + B}$.

Using logic operations in a mathematical form enables the most complex logic to be designed, simplified and converted into circuit diagram form in a very efficient and rapid way. There are more logic operators than the five we have considered here but, in general, they can all be broken down into combinations of these gates alone!